Octagonal Domains for Continuous Constraints

Marie Pelleau, Charlotte Truchet, Frédéric Benhamou

Octagonal Domains for Continuous Constraints

Marie Pelleau, Charlotte Truchet, Frédéric Benhamou

LINA, UMR CNRS 6241
Université de Nantes, France
firstName.lastName@univ-nantes.fr

Abstract. Domains in Continuous Constraint Programming (CP) are generally represented with intervals whose n-ary Cartesian product (box) approximates the solution space. This paper proposes a new representation for continuous variable domains based on octagons. We generalize local consistency and split to this octagon representation, and we propose an octagonal-based branch and prune algorithm. Preliminary experimental results show promising performance improvements on several classical benchmarks.

1 Introduction

Continuous Constraint Programming (CP) relies on interval representation of the variables domains. Filtering and solution set approximations are based on Cartesian products of intervals, called boxes. In this paper, we propose to improve the Cartesian representation precision by introducing an n-ary octagonal representation of domains in order to improve filtering accuracy.

By introducing non-Cartesian representations for domains, we do not modify the basic principles of constraint solving. The main idea remains to reduce domains by applying constraint propagators that locally approximate constraint and domains intersections (filtering), by computing fixpoints of these operators (propagation) and by splitting the domains to search the solution space. Nevertheless, each of these steps has to be redesigned in depth to take the new domains into account, since we lose the convenient correspondence between approximate intersections and domain projections.

While shifting from a Cartesian to a relational approach, the resolution process is very similar. In the interval case, one starts with the Cartesian product of the initial domains and propagators reduce this global box until reaching a fixpoint. In the octagonal case, the Cartesian product of the initial domains is itself an octagon and each constraint propagator computes in turn the smallest octagon containing the intersection of the global octagon and the constraint itself, until reaching an octagonal fixpoint. In both cases, splitting operators drive the search space exploration, alternating with global domain reduction.

The octagon are chosen for different reasons: they represent a reasonable tradeoff between boxes and more complex approximation shapes (e.g. polyhedron, ellipsoids) and they have been studied in another context to approximate numerical computations in static analysis of programs. More importantly, we

show that octagons allows us to translate the corresponding constraint systems in order to incorporate classical continuous constraint tools in the resolution.

The contributions of this paper concern the different aspects of octagon-based solving. First, we show how to transform the initial constraint problem to take the octagonal domains into account. The main idea here is to combine classical constraint matrix representations and rotated boxes, which are boxes defined in different $\pi/4$ rotated bases. Second, we define a specific local consistency, oct-consistency, and propose an appropriate algorithm, built on top of any continuous filtering method. Third, we propose a split algorithm and a notion of precision adapted to the octagonal case.

After some preliminary notions on continuous CSPs and octagons (Section 2), we present in Section 3 the octagon representation and the notion of octagonal CSP. Section 4 addresses octagonal consistency and propagation. The general solver, including discussions on split and precision is presented in Section 5. Finally, experimental results are presented in Section 6, related work in Section 7, while conclusion and future work end the presentation of this work.

2 Preliminaries

This section recalls basic notions of CP and gives material on octagons from [9].

2.1 Notations and Definitions

We consider a Constraint Satisfaction Problem (CSP) on variables $\mathcal{V} = (v_1...v_n)$, taking their values in domains $\mathcal{D} = (D_1...D_n)$, with constraints $(C_1...C_p)$. The set of tuples representing the possible assignments for the variables is $D = D_1 \times ... \times D_n$. The solutions of the CSP are the elements of D satisfying the constraints. We denote by $\mathcal{S}$ the set of solutions, $\mathcal{S} = \{(s_1...s_n) \in \mathcal{D}, \forall i \in 1..n, C_i(s_1...s_n)\}$.

In the CP framework, variables can either be discrete or continuous. In this article, we focus on real variables. Domains are subintervals of $\mathbb{R}$ whose bounds are floating points, according to the norm IEEE 754. Let $\mathbb{F}$ be the set of floating points. For $a, b \in \mathbb{F}$, we can define $[a, b] = \{x \in \mathbb{R}, a \leq x \leq b\}$ the real interval delimited by a and b, and $\mathbb{I} = \{[a, b], a, b \in \mathbb{F}\}$ the set of all such intervals. Given an interval $I \in \mathbb{I}$, we write $\underline{I}$ (resp. $\overline{I}$) its lower (resp. upper) bound, and, for any real point x, $\underline{x}$ its floating-point lower approximation (resp. $\overline{x}$, upper). A cartesian product of intervals is called a box. For CSPs with domains in $\mathbb{I}$, constraint solver usually return a box containing the solutions, that is, an overapproximation for $\mathcal{S}$.

The notion of *local consistency* is central in CP. We recall here the definition of Hull-consistency, one of the most usual local consistency for continuous constraints.

Definition 1 (Hull-Consistency). *Let $v_1...v_n$ be variables over continuous domains represented by intervals $D_1...D_n \in \mathbb{I}$, and C a constraint. The domains $D_1...D_n$ are said Hull-consistent for C iff $D_1 \times ... \times D_n$ is the smallest floating-point box containing the solutions for C.*

Given a constraint C over domains $D_1...D_n$, an algorithm that computes the local consistent domains $D'_1...D'_n$, such that $\forall i \in 1...n, D'_i \subset D_i$ and $D'_1...D'_n$ are locally consistent for C, is called a *propagator* for C. The domains which are locally consistent for all constraints are the largest common fixpoints of all the constraints propagators [2, 12]. Practically, propagators often compute overapproximations of the locally consistent domains. We will use the standard algorithm HC4 [3]. It efficiently propagates continuous constraints, relying on the syntax of the constraint and interval arithmetic [10]. It generally does not reach Hull consistency, in particular in case of multiple occurrences of the variables.

Local consistency computations can be seen as deductions, performed on domains by analyzing the constraints. If the propagators return the empty set, the domains are inconsistent and the problem has no solution. Otherwise, non-empty local consistent domains are computed. This is often not sufficient to accurately approximate the solution set. In that case choices are made on the variables values. For continuous constraints, a domain D is chosen and split into two (or more) parts, which are in turn narrowed by the propagators. The solver recursively alternates propagations and splits until a given precision is reached. In the end, the collection of returned boxes covers $\mathcal{S}$, under some hypotheses on the propagators and splits [2].

2.2 Octagons

In geometry, an octagon is a polygon having eight faces in $\mathbb{R}^2$ [1]. In this paper, we use a more general definition given in [9].

Definition 2 (Octagonal constraints). *Let v_i, v_j be two real variables. An octagonal constraint is a constraint of the form $\pm v_1 \pm v_2 \leq c$ with $c \in \mathbb{R}$.*

For instance in $\mathbb{R}^2$, octagonal constraints define straight lines which are parallel to the axis if $i = j$, and diagonal if $i \neq j$. This remains true in $\mathbb{R}^n$, where the octagonal constraints define hyperplanes.

Definition 3 (Octagon). *Given a set of octagonal constraints $\mathcal{O}$, the subset of $\mathbb{R}^n$ points satisfying all the constraints in $\mathcal{O}$ is called an octagon.*

Remark 1. Here follows some general remarks on octagons :

- The geometric shape defined above includes the geometric octagons, but also other polygons (e.g. in $\mathbb{R}^2$, an octagon can have less than eight faces);
- an octagon can be defined with redundant constraints (for example $v_1 - v_2 \leq c$ and $v_1 - v_2 \leq c'$), but only one of them defines a face of the octagon (the one with the lowest constant in this example),
- in $\mathbb{R}^n$, an octagon has at most $2n^2$ faces, which is the maximum number of possible non-redundant octagonal constraints on n variables,
- an octagon is a set of *real* points, but, like the intervals, they can be restricted to have floating-points bounds ($c \in \mathbb{F}$).

In the following, octagons are restricted to floating-point octagons. Without loss of generality, we assume octagons to be defined with no redundancies.

[1] http://mathworld.wolfram.com/Octagon.html

2.3 Matrix Representation of Octagons

An octagon can be represented with a *difference bound matrix* (DBM) as described in [8, 9]. This representation is based on a normalization of the octagonal constraints as follows.

Definition 4 (difference constraints). *Let w, w' be two variables. A difference constraint is a constraint of the form $w - w' \leq c$ for c a constant.*

By introducing new variables, it is possible to rewrite an octagonal constraint as an equivalent difference constraint: let $C \equiv (\pm v_i \pm v_j \leq c)$ an octagonal constraint. Define the new variables $w_{2i-1} = v_i, w_{2i} = -v_i, w_{2j-1} = v_j, w_{2j} = -v_j$. Then

- for $i \neq j$
  - if $C \equiv (v_i - v_j \leq c)$, then C is equivalent to the difference constraints $(w_{2i-1} - w_{2j-1} \leq c)$ and $(w_{2j} - w_{2i} \leq c)$,
  - if $C \equiv (v_i + v_j \leq c)$, then C is equivalent to the difference constraints $(w_{2i-1} - w_{2j} \leq c)$ and $(w_{2j-1} - w_{2i} \leq c)$,
  - the two other cases are similar,
- for $i = j$
  - if $C \equiv (v_i - v_i \leq c)$, then C is pointless, and can be removed,
  - if $C \equiv (v_i + v_i) \leq c)$, then C is equivalent to the difference constraint $(w_{2i-1} - w_{2i} \leq c)$,
  - the two other cases are similar.

In what follows, regular variables are always written $(v_1 ... v_n)$, and the corresponding new variables are written $(w_1, w_2, ... w_{2n})$ with: $w_{2i-1} = v_i$, and $w_{2i} = -v_i$. As shown in [9], the rewritten difference constraints represent the same octagon as the original set of octagonal constraints, by replacing the positive and negative occurrences of the v_i variables by their w_i counterparts. Storing difference constraints is thus a suitable representation for octagons.

Definition 5 (DBM). *Let $\mathcal{O}$ be an octagon in $\mathbb{R}^n$, and its sequence of potential constraints as defined above. The octagon DBM is a $2n \times 2n$ square matrix, such that the element at row i, column j is the constant c of the potential constraint $w_j - w_i \leq c$.*

An example is shown on Figure 1(c): the element on row 1 and column 3 corresponds to the constraint $v_2 - v_1 \leq 2$ for instance.

At this stage, different DBMs can represent the same octagon. For example on Figure 1(c), the element row 2 and column 3 can be replaced with 100, for instance, without changing the corresponding octagon. In [9], an algorithm is defined so as to optimally compute the smallest values for the elements of the DBM. This algorithm is adapted from the Floyd-Warshall shortest path algorithm [6], modified in order to take advantage of the DBM structure. It exploits the fact that w_{2i-1} and w_{2i} correspond to the same variable. It is fully presented in [9].

3 Boxes Representation

In the following section we introduce a box representation for octagons. This representation, combined with the DBM will be used to define, from an initial set of continuous constraints, an equivalent system taking the octagonal domains into account.

3.1 Intersection of boxes

In two dimensions, an octagon can be represented by the intersection of one box in the canonical basis for $\mathbb{R}^2$, and one box in the basis obtained from the canonical basis by a rotation of angle $\pi/4$ (see figure 1(b)). We generalize this remark to n dimensions.

Definition 6 (Rotated basis). *Let $B = (u_1, ..., u_n)$ be the canonical basis of $\mathbb{R}^n$. Let $\alpha = \pi/4$. The* (i,j)-rotated basis $B_\alpha^{i,j}$ *is the basis obtained after a rotation of α in the subplane defined by (u_i, u_j), the other vectors remaining unchanged: $B_\alpha^{i,j} = (u_1, ..., u_{i-1}, (cos(\alpha)u_i + sin(\alpha)u_j), ...u_{j-1}, (-sin(\alpha)u_i + cos(\alpha)u_j), ...u_n)$.*

By convention, for any $i \in \{1...n\}$, $B_\alpha^{i,i}$ represents the canonical basis. In what follows, α is always $\pi/4$ and will be omitted. Finally, every variable v living in the $B^{i,j}$ rotated basis and whose domain is D will be denoted by $v^{i,j}$ and its domain by $D^{i,j}$).

The DBM can also be interpreted as the representation of the intersection of one box in the canonical basis and $n(n-1)/2$ other boxes, each one living in a rotated basis. Let $\mathcal{O}$ be an octagon in $\mathbb{R}^n$ and its DBM M, with the same notations as above (M is a $2n \times 2n$ matrix). For $i, j \in \{1...n\}$, with $i \neq j$, let $\mathcal{B}_\mathcal{O}^{i,j}$ be the box $I_1 \times ... \times I_i^{i,j} \times ... \times I_j^{i,j} ... \times I_n$, in $B^{i,j}$, such that $\forall k \in \{1...n\}$

$$\begin{array}{ll} \underline{I_k} = \underline{-\frac{1}{2}M[2k-1, 2k]} & \overline{I_k} = \overline{\frac{1}{2}M[2k, 2k-1]} \\ \underline{I_i^{i,j}} = \underline{-\frac{1}{\sqrt{2}}M[2j-1, 2i]} & \overline{I_i^{i,j}} = \overline{\frac{1}{\sqrt{2}}M[2j, 2i-1]} \\ \underline{I_j^{i,j}} = \underline{-\frac{1}{\sqrt{2}}M[2j-1, 2i-1]} & \overline{I_j^{i,j}} = \overline{\frac{1}{\sqrt{2}}M[2j, 2i]} \end{array}$$

Proposition 1. *Let $\mathcal{O}$ be an octagon in $\mathbb{R}^n$, and $\mathcal{B}_\mathcal{O}^{i,j}$ the boxes as defined above. Then $\mathcal{O} = \bigcap_{1 \leq i,j \leq n} \mathcal{B}_\mathcal{O}^{i,j}$.*

Proof. Let $i, j \in \{1..n\}$. We have $v_i^{i,j} = 1/\sqrt{2}(v_i + v_j)$ and $v_j^{i,j} = 1/\sqrt{2}(v_i - v_j)$ by definition 6. Thus $(v_1...v_i^{i,j}...v_j^{i,j}...v_n) \in \mathcal{B}_\mathcal{O}^{i,j}$ iff it satisfies the octagonal constraints on v_i and v_j, and the unary constraints for the other coordinates, in the DBM. The box $\mathcal{B}_\mathcal{O}^{i,j}$ is thus the solution set for these particular octagonal constraints. The points in $\bigcap_{1 \leq i,j \leq n} \mathcal{B}_\mathcal{O}^{i,j}$ are exactly the points which satisfy all the octagonal constraints.

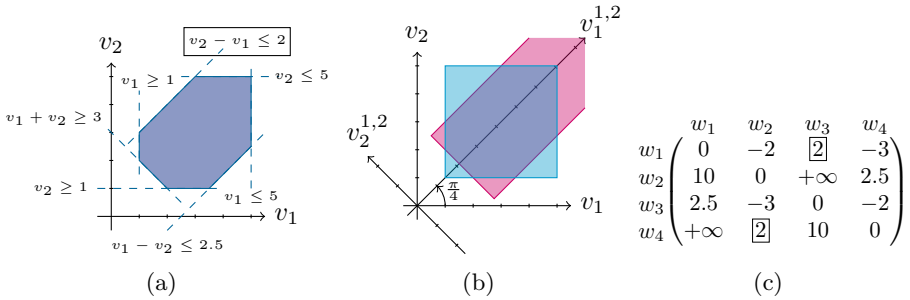

Fig. 1. Equivalent representations for the same octagon: the octagonal constraints 1(a), the intersection of boxes 1(b), and the DBM 1(c).

Example 1. Considering the DBM Figure 1(c), the boxes are $I_1 \times I_2 = [1, 5] \times [1, 5]$, and $I_1^{1,2} \times I_2^{1,2} = [3/\sqrt{2}, +\infty] \times [-2.5/\sqrt{2}, \sqrt{2}]$.

To summarize, an octagon with its DBM representation can also be interpreted as a set of octagonal constraints (definition in intension) or equivalently as an intersection of rotated boxes (definition in extension), at the cost of a multiplication / division with the appropriate rounding mode. We show below that the octagonal constraints (or the bounds in the case of boxes) can be inferred from the CSP.

3.2 Octagonal CSP

Consider a CSP on variables $(v_1...v_n)$ in $\mathbb{R}^n$. The goal is now to equip this CSP with an octagonal domain. We detail here how to build an octagonal CSP from a regular one, and show that the two systems are equivalent.

First, the CSP is associated to an octagon, by stating all the possible octagonal constraints $\pm v_i \pm v_j \leq c_k$ for $i, j \in \{1...n\}$. The constants c_k represent the bounds of the octagon boxes and are dynamically modified. They are initialized to $+\infty$.

The rotations defined in the previous section introduce new axes, that is, new variables $v_i^{i,j}$. Because these variables are redundant with the regular ones, they are also linked through the CSP constraints $(C_1...C_p)$, and these constraints have to be rotated as well.

Given a function f on variables $(v_1...v_n)$ in B, the expression of f in the (i, j)-rotated basis is obtained by symbolically replacing the i-th and j-th coordinates by their expressions in $B_\alpha^{i,j}$. Precisely, replace v_i by $\cos(\alpha)v_i^{i,j} - \sin(\alpha)v_j^{i,j}$ and v_j by $\sin(\alpha)v_i^{i,j} + \cos(\alpha)v_j^{i,j}$ where $v_i^{i,j}$ and $v_j^{i,j}$ are the coordinates for v_i and v_j in $B_\alpha^{i,j}$. The other variables are unchanged.

Definition 7 (Rotated constraint). *Given a constraint C_k holding on variables $(v_1...v_n)$, the (i, j)-rotated constraint $C_k^{i,j}$ is the constraint obtained by re-*

placing each occurrence of v_i *by* $\cos(\alpha)v_i^{i,j} - \sin(\alpha)v_j^{i,j}$ *and each occurrence of* v_j *by* $\sin(\alpha)v_i^{i,j} + \cos(\alpha)v_j^{i,j}$.

Given a continuous CSP $< v_1...v_n, D_1...D_n, C_1...C_p >$, we define an octagonal CSP by adding the rotated variables, the rotated constraints, and the rotated domains stored as a DBM. To sum up and fix the notations, the octagonal CSP thus contains:

- the regular variables $(v_1...v_n)$;
- the rotated variables $(v_1^{1,2}, v_2^{1,2}, v_1^{1,3}, v_3^{1,3}...v_n^{n-1,n})$, where $v_i^{i,j}$ is the i-th variable in the (i,j) rotated basis $B_\alpha^{i,j}$;
- the regular constraints $(C_1...C_p)$;
- the rotated constraints $(C_1^{1,2}, C_1^{1,3}...C_1^{n-1,n}...C_p^{n-1,n})$.
- the regular domains $(D_1...D_n)$;
- a DBM which represents the rotated domains. It it initialized with the bounds of the regular domains for the cells at position $2i, 2i-1$ and $2i-1, 2i$ for $i \in \{1...2n\}$, and $+\infty$ everywhere else.

In these conditions, the initial CSP is equivalent to this transformed CSP, restricted to the variables $v_1...v_n$, as shown in the following proposition.

Proposition 2. *Consider a CSP* $< v_1...v_n, D_1...D_n, C_1...C_p >$*, and the corresponding octagonal CSP as defined above. The solution set of the original CSP* $\mathcal{S}$ *is equal to the solution set of the* $(v_1...v_n)$ *variables of the octagonal CSP.*

Proof. Let $s \in \mathbb{R}^n$ a solution of the octogonal CSP for $(v_1...v_n)$. Then $s \in D_1 \times ... \times D_n$ and $C_1(s)...C_p(s)$ are all true. Hence s is a solution for the original CSP. Reciprocally, let $s' \in \mathbb{R}^n$ a solution of the original CSP. The regular constraints $(C_1...C_p)$ are true for s'. Let us show that there exist values for the rotated variables such that the rotated constraints are true for s'. Let $i, j \in \{1...n\}$, $i \neq j$, and $k \in \{1...p\}$ and $C_k^{i,j}$ the corresponding rotated constraint. By definition 7, $C_k^{i,j}(v_1...v_{i-1}, \cos(\alpha)v_i^{i,j} - \sin(\alpha)v_j^{i,j}, v_{i+1}... \sin(\alpha)v_i^{i,j} + \cos(\alpha)v_j^{i,j}...v_n) \equiv C_k(v_1...v_n)$. Let us define the two reals $s_i^{i,j} = cos(\alpha)s_i + sin(\alpha)s_j$ and $s_j^{i,j} = -sin(\alpha)s_i + cos(\alpha)s_j$, the image of s_i and s_j by the rotation of angle α. By reversing the rotation, $cos(\alpha)s_i^{i,j} - sin(\alpha)s_j^{i,j} = s_i$ and $sin(\alpha)s_i^{i,j} + cos(\alpha)s_j^{i,j} = s_j$, thus $C_k^{i,j}(s_1...s_i^{i,j}, ...s_j^{i,j}...s_n) = C_k(s_1...s_n)$ is true. It remains to check that $(s_1...s_i^{i,j}, ...s_j^{i,j}...s_n)$ is in the rotated domain, which is true because the DBM is initialized at $+\infty$. □

For a CSP on n variables, this representation has an order of magnitude n^2, with n^2 variables and domains, and $p\frac{n(n-1)}{2} + p$ constraints. Of course, many of these objects are redundant. We explain in the next sections how to use this redundancy to speed up the solving process.

4 Octagonal Consistency and Propagation

We first generalize the Hull-consistency definition to the octagonal domains, and define propagators for the rotated constraints. Then, we use the modified version of Floyd-Warshall (briefly described in section 2.3) to define an efficient propagation scheme for both octagonal and rotated constraints.

4.1 Octagonal Consistency for a Constraint

We generalize to octagons the definition of Hull-consistency on intervals for any continuous constraint. With the box representation, we show that any propagator for Hull-consistency on boxes can be extended to a propagator on the octagons. For a given n-ary relation on $\mathbb{R}^n$, there is a unique smallest octagon (wrt inclusion) which contains the solutions of this relation, as shown in the following proposition.

Remark 2. Consider a constraint C (resp. a constraint sequence $(C_1...C_p)$), and $\mathcal{S}_C$ its set of solutions (resp. $\mathcal{S}$). Then there exist a unique octagon $\mathcal{O}$ such that: $\mathcal{S}_C \subset \mathcal{O}$ (resp. $\mathcal{S} \subset \mathcal{O}$), and for all octagons $\mathcal{O}'$, $\mathcal{S}_C \subset \mathcal{O}'$ implies $\mathcal{O} \subset \mathcal{O}'$. $\mathcal{O}$ is the unique smallest octagon containing the solutions, wrt inclusion.

Definition 8 (Oct-Consistency). *Consider a constraint C (resp. a constraint sequence $(C_1...C_p)$), and $\mathcal{S}_C$ its set of solutions (resp. $\mathcal{S}$). An octagon is said Oct-consistent for this constraint iff it is the smallest octagon containing $\mathcal{S}_C$ (resp. $\mathcal{S}$), wrt inclusion.*

From remark 2, this definition is well founded. With the expression of an (i, j)-rotated constraint $C^{i,j}$, any propagator defined on the boxes can be used to compute the Hull-consistent boxes for $C^{i,j}$ (although such a propagator, as HC4, may not reach consistency). This gives a consistent box in basis $B^{i,j}$, and can be done for all the bases. The intersection of the Hull-consistent boxes is the Hull-consistent octagon.

Proposition 3. *Let C be a constraint, and $i, j \in \{1...n\}$. Let $\mathcal{B}^{i,j}$ be the Hull-consistent box for the rotated constraint $C^{i,j}$, and $\mathcal{B}$ the Hull-consistent box for C. The Oct-consistent octagon for C is the intersection of all the $\mathcal{B}^{i,j}$ and $\mathcal{B}$.*

Proof. Let $\mathcal{O}$ be the Oct-consistent octagon. By definition 2, a box is an octagon. Since $\mathcal{O}$ is the smallest octagon containing the solutions, and all the $\mathcal{B}^{i,j}$ contain the solutions, for all $i, j \in \{1...n\}$, $i \neq j$ $\mathcal{O} \subset \mathcal{B}^{i,j}$ (the same holds for $\mathcal{B}$). Thus $\mathcal{O} \subset \bigcap_{1 \leq i,j \leq n} \mathcal{B}^{i,j}$. Reciprocally, we use the box representation for the Oct-consistent octagon: $\mathcal{O} = \bigcap_{1 \leq i,j \leq n} \mathcal{B}_o^{i,j}$, where $\mathcal{B}_o^{i,j}$ is the box defining the octagon in $B^{i,j}$. Because $\mathcal{O}$ contains all the solutions and $\mathcal{B}_o^{i,j}$ contains $\mathcal{O}$, $\mathcal{B}_o^{i,j}$ also contains all the solutions. Since $\mathcal{B}^{i,j}$ is the Hull-consistent box in $B_\alpha^{i,j}$, it is the smallest box in $B_\alpha^{i,j}$ which contains all the solutions. Thus $\mathcal{B}^{i,j} \subset \mathcal{B}_o^{i,j}$. From

```
float dbm[2n, 2n] /*the dbm containing the octagonal constraints*/
list propagList ← (ρ_{C_1}, ...ρ_{C_p}, ρ_{C_1^{1,2}}...ρ_{C_p^{n-1,n}}) /*list of the propagators to apply*/
while propagList ≠ ∅ do
  apply all the propagators of propagList to dbm /*initial propagation*/
  propagList ← ∅
  for i,j from 1 to n do
    m ← minimum of (dbm[2i − 1, k]+dbm[k, 2j − 1]) for k from 1 to 2n
    m ← minimum(m, dbm[2i − 1, 2i]+dbm[2j, 2j − 1])
    if m < dbm[2i − 1, 2j − 1] then
      dbm[2i − 1, 2j − 1] ← m /*update of the DBM*/
      add ρ_{C_1^{i,j}}...ρ_{C_p^{i,j}} to propagList /*get the propagators to apply*/
    end if
    repeat the 5 previous steps for dbm[2i−1, 2j], dbm[2i, 2j−1], and dbm[2i, 2j]
  end for
end while
```

Fig. 2. Pseudo code for the propagation loop mixing the Floyd Warshall algorithm (the *for* loop) and the regular and rotated propagators $\rho_{C_1}...\rho_{C_p}, \rho_{C_1^{1,2}}...\rho_{C_p^{n-1,n}}$, for an octagonal CSP as defined in section 3.2.

there, $\bigcap_{1\leq i,j\leq n} \mathcal{B}^{i,j} \subset \bigcap_{1\leq i,j\leq n} \mathcal{B}_o^{i,j} = \mathcal{O}$. Again, the same holds for $\mathcal{B}$. The two inclusions being proven, $\mathcal{O} = \bigcap_{1\leq i,j\leq n} \mathcal{B}^{i,j}$. □

4.2 Propagation Scheme

The propagation scheme presented in subsection 2.3 for the octagonal constraints is optimal. We thus rely on this propagation scheme, and integrate the non-octagonal constraints propagators in this loop. The point is to use the octagonal constraints to benefit from the relational properties of the octagon. This can be done thanks to the following remark: all the propagators defined in the previous subsections are monotonic and complete (as is the HC4 algorithm). It results that they can be combined in any order in order to achieve consistency, as shown for instance in [2].

The key idea for the propagation scheme is to interleave the refined Floyd-Warshall algorithm and the constraint propagators. A pseudocode is given on figure 2. At the first level, the DBM is recursively visited so that the minimal bounds for the rotated domains are computed. Each time a DBM cell is modified, the corresponding propagators are added to the propagation list. The propagation list is applied before each new round in the DBM (so that a cell that would be modified twice is propagated only once). The propagation is thus guided by the additional information of the relational domain. This is illustrated on Figure 3: the propagators $\rho_{C_1}...\rho_{C_p}, \rho_{C_1^{1,2}}...\rho_{C_p^{n-1,n}}$ are first applied (3(a), 3(b)), then the boxes are made consistent wrt each other using the refined Floyd-Warshall algorithm.

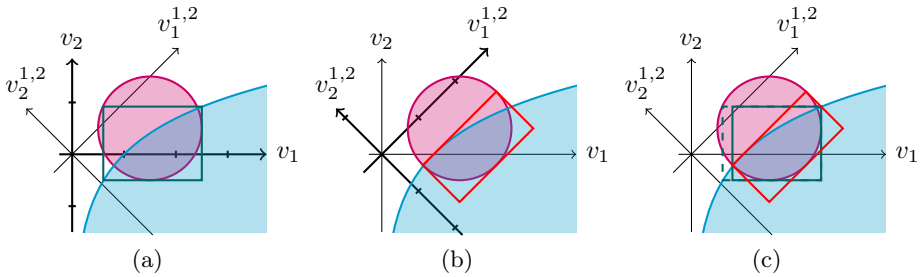

Fig. 3. Example of the Oct-consistency: an usual consistency algorithm is applied in each basis (Figures 3(a) and 3(b)) then the different boxes are made consistent using the modified Floyd-Warshall algorithm (Figure 3(c)).

We show here that the propagation as defined on figure 2 computes the consistent octagon for a sequence of constraints.

Proposition 4 (Correctness). *Let* $< v_1...v_n, D_1...D_n, C_1...C_p >$ *a CSP. Assume that, for all* $i, j \in \{1...n\}$*, there exists a propagator* ρ_C *for the constraint* C*, such that* ρ_C *reaches Hull consistency, that is,* $\rho_C(D_1 \times ... \times D_n)$ *is the Hull consistent box for* C*. Then the propagation scheme as defined on figure 2 computes the Oct-consistent octagon for* $C_1...C_p$*.*

Proof. This derives from proposition 3, and the propagation scheme of figure 2. The propagation scheme is defined so as to stop when propagList is empty. This happens when $\forall i, j \in \{1...n\}, k \in \{1...2n\}$, $\text{dbm}[2i-1,k]+\text{dbm}[k,2j-1]$, $\text{dbm}[2i-1,2i]+\text{dbm}[2j,2j-1] \geq \text{dbm}[2i-1,2j-1]$, the same holds for $\text{dbm}[2i-1,2j]$, $\text{dbm}[2i,2j-1]$, and $\text{dbm}[2i,2j]$. The octagonal constraints are thus consistent. In addition, each time a rotated box is modified in the DBM, its propagators are added to propagList. Hence, the final octagon is stable by the application of all $\rho_{C_k^{i,j}}$, for all $k \in \{1...p\}$ and $i, j \in \{1...n\}$. By hypothesis, the propagators reach consistency, the boxes are thus Hull-consistent for all the (rotated and regular) constraints. By proposition 3, the returned octagon is Oct-consistent. □

The refined Floyd-Warshall has a time complexity of $O(n^3)$. For each round in its loop, in the worst case we add p propagators in the propagation list. Thus the time complexity for the propagation scheme of figure 2 is $O(n^3p^3)$. In the end, the octagonal propagation uses both representations of octagons. It takes advantage of the relational property of the octagonal constraints (Floyd-Warshall), and of the usual constraint propagation on boxes (propagators). This comes to the cost of computing the octagon, but is expected to give a better precision in the end.

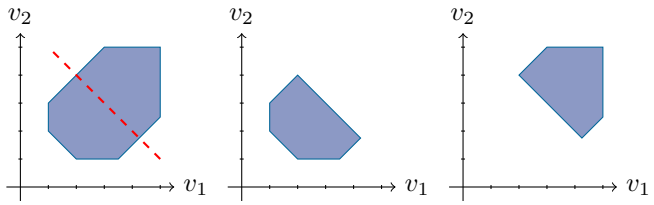

Fig. 4. Example of a split: the octagon on the left is cut in the $B^{1,2}$ basis.

5 Solving

Besides the expected gain in precision obtained with octagon consistency, the box representation of octagons allows us to go a step further and define a fully octagonal solver. We thus define an octagonal split, in order to be able to cut the domains in any octagonal direction, and an octagonal precision, and end up with a fully octagonal solver.

5.1 Octagonal Split

The octagonal split extends the usual split operator to octagons. Splits can be performed in the canonical basis, thus being equivalent to the usual splits, or in the rotated basis. It can be defined as follow:

Definition 9. *Given an octagonal domain defined with the box representation* $D_1...D_n, D_1^{1,2}...D_1^{n-1,n}...D_n^{n-1,n}$*, such that* $D_k^{i,j} = [a, b]$*, a splitting operator for variable* $v_k^{i,j}$*, computes the two octagonal subdomains* $D_1...[a, \overline{(a+b)/2}]...D_n^{n-1,n}$ *and* $D_1...[\underline{(a+b)/2}, b]...D_n^{n-1,n}$*.*

As for the usual split, the union of the two octagonal subdomains is the original octagon, thus the split does not lose solutions. This definition does not take into account the correlation between the variables of the different basis. We take advantage again of the octagonal representation to communicate the domain reduction to the other basis. A split is thus immediately followed by a Floyd-Warshall propagation. Figure 4 shows an example of the split.

5.2 Precision

In most continuous solvers, the precision is defined as the size of the largest domain. For octagons, this definition leads to a loss of information because it does not take into account the correlation between the variables and domains.

Definition 10. *Let* $\mathcal{O}$ *be an octagon, and* $I_1...I_n, I_1^{1,2}...I_n^{n-1,n}$ *its box representation. The octagonal precision is* $\tau(\mathcal{O}) = min_{1\leq i,j\leq n}(max_{1\leq k\leq n}(\overline{I_k^{i,j}} - \underline{I_k^{i,j}}))$*.*

```
Octogone oct
queue splittingList ← oct /*queue of the octagons*/
list acceptedOct ← ∅ /*list of the accepted octagons*/
while splittingList ≠ ∅ do
   Octogone octAux ← splittingList.top()
   splittingList.pop()
   octAux ← Oct-consistence(octAux)
   if τ_Oct(octAux) < r or octAux contains only solutions then
      add octAux to acceptedOct
   else
      Octogone leftOct ← left(octAux) /*left and right are the split operators*/
      Octogone rightOct ← right(octAux)
      add leftOct to splittingList
      add rightOct to splittingList
   end if
end while
return acceptedOct
```

Fig. 5. Solving with octagons.

For a single regular box, τ would be the same precision as usual. On an octagon, we take the minimum precision of the boxes in all the bases because it is more accurate, and it allows us to retrieve the operational semantics of the precision, as shown by the following proposition: in an octagon of precision r overapproximating a solution set $\mathcal{S}$, every point is at a distance at most r from $\mathcal{S}$.

Proposition 5. *Let $< v_1...v_n, D_1...D_n, C_1...C_p >$ be a CSP, and $\mathcal{O}$ an octagon overapproximating $\mathcal{S}$. Let $r = \tau(\mathcal{O})$. Let $(v_1, ...v_n) \in \mathbb{R}^n$ be a point of $D_1 \times ... \times D_n$. Then $\forall 1 \leq i \leq n$, $min_{s \in \mathcal{S}} |v_i - s_i| \leq r$, where $s = (s_1...s_n)$. Each coordinate of all the points in $\mathcal{O}$ are at a distance at most r of a solution.*

Proof. By definition 10, the precision r is the minimum of some quantities in all the rotated basis. Let $B^{i,j}$ be the basis that realizes this minimum. Because the box $\mathcal{B}^{i,j} = D_1 \times ... \times D_i^{i,j} \times ...D_j^{i,j} ... \times D_n$ is Hull-consistent by proposition 3, it contains $\mathcal{S}$. Let $s \in \mathcal{S}$. Because $r = max_k(\overline{D_k} - \underline{D_k})$, $\forall 1 \leq k \leq n$, $|s_k - v_k| \leq \overline{D_k} - \underline{D_k} \leq r$. □

5.3 Octagonal Solver

Figure 5 describes the octagonal solving process. By proposition 4, and the split property, it returns a sequence of octagons whose union overapproximate the solution space. Precisely, it returns either octagons for which all points are solutions, or octagons overapproximating solution sets with a precision r.

An important feature of a constraint solver is the variable heuristic. For continuous constraints, one usually choose to split the variable that has the largest domain. This would be very bad for the octagons, as the variable which has the

largest domain is probably in a basis that is of little interest for the problem (it probably has a wide range because the constraints are poorly propagated in this basis). We thus define a default octagonal strategy which relies on the same remark as for definition 10: the variable to split is the variable $V_k^{i,j}$ which realizes the minimum of $min_{1\leq i,j\leq n}(max_{1\leq k\leq n}(\overline{D_k^{i,j}} - \underline{D_k^{i,j}}))$. The strategy is the following: choose first a promising basis, that is, a basis in which the boxes are tight (choose i, j). Then take the worst variable in this basis as usual (choose k).

6 Experiments

This section compares the octagonal solver with a traditional interval solver on classical benchmarks.

6.1 Implementation

We have implemented a prototype of the octagonal solver, with Ibex, a C++ library for continuous constraints [4]. We use the Ibex implementation of *HC4-Revise* [3] to contract the constraints. The octagons are implemented with their DBM representation. Additional rotated variables and constraints are posted and dealt with as explained above.

An important point is the rotation of the constraints. The HC4 algorithm is sensitive to multiple occurrences of the variables, and the symbolic rewriting defined in section 3.2 creates multiple occurrences. Thus, the HC4 propagation on the rotated constraints could be very poor if performed directly on the rotated constraints. It is necessary to simplify the rotated constraints wrt the number of multiple occurrences for the variables. We use the Simplify function of Mathematica to do this. The computation time indicated below does not include the time for this treatment, however, it is negligible compared to the solving times. The propagator is an input of our method: we used a standard one (HC4), but more recent propagator such as [1] will be considered in the future. It is sufficient for our needs that the consistency algorithms computes overapproximations of the Hull-consistent boxes, as it is often the case for continuous propagators.

6.2 Results

We have tested the prototype octagonal solver on problems from the Coconut benchmark[2]. These problems have been chosen depending on the type of the constraints (inequations, equations, or both).

Experiments have been made, with Ibex 1.18, on a MacBook Pro Intel Core 2 Duo 2.53 GHz. Apart from the variable heuristic presented in subsection 5.3, the experiments have been done with the same configuration in Ibex, in particular, using the same propagators, so as to compare exactly the octagonal results

[2] This benchmark can be found at http://www.mat.univie.ac.at/ neum/glopt/coconut/

	First solution				All the solutions			
	$\mathbb{I}^n$		Oct		$\mathbb{I}^n$		Oct	
h75 5 ≤	41.40	*1* 1 024 085	**0.03**	*1* **149**	> 3 hours		> 3 hours	
hs64 3 ≤	**0.01**	*1* 217	0.05	*1* **67**	> 3 hours		> 3 hours	
h84 5 ≤	5.47	*1* 87 061	**2.54**	*1* **1 407**	> 3 hours		**7238.74**	***10 214 322*** **22 066 421**
KinematicPair 2 ≤	**0.00**	*1* 45	**0.00**	*1* **23**	53.09	*424 548* 893 083	**16.56**	***39 555*** **79 125**
pramanik 3 =	28.84	*1* 321 497	**0.16**	*1* **457**	**193.14**	***145 663*** 2 112 801	543.46	*210 371* **1 551 157**
trigo1 10 =	18.93	*1* 10 667	**1.38**	*1* **397**	**20.27**	***12*** 11 137	28.84	*347* **5 643**
brent-10 10 =	6.96	*1* 115 949	**0.54**	*1* **157**	**17.72**	*854* 238 777	105.02	***142*** **100 049**
h74 5 = ≤	305.98	*1* 8 069 309	**13.70**	*1* **138 683**	1 304.23	***183 510*** 20 061 357	**566.31**	*700 669* **1 926 455**
fredtest 6 = ≤	3 146.44	*1* 29 206 815	**19.33**	*1* **3 281**	> 3 hours		> 3 hours	

Table 1. Results on problems from the Coconut benchmark. The first column gives the name of the problem, the number of variable and the type of the constraints. In each cell, the number on the left is the CPU time in seconds. Upper right is the number of box in the computed solution, lower right the number of created boxes.

with their interval counterparts. Table 1 compares the results obtained by the interval resolution to those obtained by the octagonal resolution. In the first four problems the constraints are inequalities, in the three following they are only equalities and in the last two they are mixed inequalities and equalities. The octagonal solver needs less time and created less boxes to find the first solution of a problem. We obtain better results on problems containing inequalities. Problems with equalities contain multiple occurrences of variables, which can explain the bad results obtained by the octagonal solver on those problems.

7 Related Works

Our work is related to [9], in static analysis of programs. Their goal is to compute overapproximations for the traces of a program. The octagons are shown to provide a good trade off between the precision of the approximation and the computation cost. We use their matrix representation and their version of the Floyd-Warshall algorithm.

Propagation algorithms for the difference constraints, also called *temporal*, have already presented in [5, 11]. They have a better complexity than the one we use, but are not suited to the DBM case, because they do not take into account the doubled variables.

The idea of rotating variables and constraints has already been proposed in [7], in order to better approximate the solution set. Their method is dedicated to under-constrained systems of equations.

8 Conclusion

In this paper, we have proposed a solving algorithm for continuous constraints based on octagonal approximations. Starting from the remark that domains in Constraint Programming can be interpreted as components of a global multi-dimensional parallelepipedic domain, we have constructed octagonal approximations on the same model and provided algorithms for octagonal CSP transformations, filtering, propagation, precision and splitting. An implementation based on Ibex and preliminary experimental results on classical benchmarks are encouraging, particularly in the case of systems containing inequalities. Future work involves the experimental study of other interval-based propagators such as Mohc [1] and extensions to other geometric structures.

References

1. Ignacio Araya, Gilles Trombettoni, and Bertrand Neveu. Exploiting monotonicity in interval constraint propagation. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence, AAAI 2010*, 2010.
2. Frédéric Benhamou. Heterogeneous constraint solvings. In *Proceedings of the 5th International Conference on Algebraic and Logic Programming*, pages 62–76, 1996.
3. Frédéric Benhamou, Frédéric Goualard, Laurent Granvilliers, and Jean-François Puget. Revisiting hull and box consistency. In *Proceedings of the 16th International Conference on Logic Programming*, pages 230–244, 1999.
4. Gilles Chabert and Luc Jaulin. Contractor programming. *Artificial Intelligence*, 173:1079–1100, 2009.
5. Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. In *Proceedings of the first international conference on Principles of Knowledge Representation and Reasoning*, 1989.
6. Robert Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6), 1962.
7. Alexandre Goldsztejn and Laurent Granvilliers. A new framework for sharp and efficient resolution of ncsp with manifolds of solutions. In *Proceedings of the 14th international conference on Principles and Practice of Constraint Programming (CP '08)*, pages 190–204, Berlin, Heidelberg, 2008. Springer-Verlag.
8. Miguel Menasche and Bernard Berthomieu. Time petri nets for analyzing and verifying time dependent communication protocols. In *Protocol Specification, Testing, and Verification*, 1983.
9. Antoine Miné. The octagon abstract domain. *Higher-Order and Symbolic Computation*, 19(1):31–100, 2006.
10. Ramon Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs N. J., 1966.
11. Jean-Charles Régin and Michel Rueher. *Inequality-sum : a global constraint capturing the objective function*. RAIRO Operations Research, 2005.
12. Christian Schulte and Peter J. Stuckey. Efficient constraint propagation engines. *Transactions on Programming Languages and Systems*, 31(1):2:1–2:43, December 2008.